

UNITED STATES PATENT APPLICATION

FOR

A METHOD AND APPARATUS FOR SUPPORTING MULTISERVICE
DIGITAL SIGNAL PROCESSING APPLICATIONS

INVENTOR:

LOUIS COUTURE

Prepared By:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP
12400 Wilshire Boulevard
Seventh Floor
Los Angeles, California 90025-1026
(408) 720-8598

Attorney's Docket No.: 81862.P082

"Express Mail" mailing label number: EL034144732US

Date of Deposit: March 26, 1998

I hereby certify that I am causing this paper or fee to be deposited with the United States Postal Service
"Express Mail Post Office to Addressee" service on the date indicated above and that this paper or fee has
been addressed to the Assistant Commissioner for Patents, Washington, D. C. 20231

Cindy Baglietto

(Typed or printed name of person mailing paper or fee)

Cindy Baglietto
(Signature of person mailing paper or fee)

3/26/98
(Date signed)

A METHOD AND APPARATUS FOR SUPPORTING MULTISERVICE DIGITAL SIGNAL PROCESSING APPLICATIONS

FIELD OF THE INVENTION

5 The present invention relates generally to a mechanism for supporting multiservice digital signal processing (DSP) applications and more particularly to the simultaneous processing of a number of data types on a host platform.

10 BACKGROUND OF THE INVENTION

 Until recently there has persisted a fundamental dichotomy between two main types of telecommunication networks. The first type of telecommunication network, the telephone network, switches and transports predominantly voice, facsimile, and modulation-demodulation system
15 (modem) traffic. The public switched telephone network (PSTN) is an example of this type of network. Telephone networks are also deployed privately within organizations such as corporations, banks, campuses, and government offices. The second type of telecommunication network, the data network, switches or routes and transports data between computers. The
20 Internet is an example of a public data network; data networks may be privately deployed.

 Telephone networks were developed and deployed earlier, followed by data networks. Telephone network infrastructures are ubiquitous, however, and as a result data networks typically are built, to a limited extent, using
25 some components of telephone networks. For example, the end user access link to a data network in some cases is implemented with a dial-up telephone line. The dial-up telephone line thus connects the end user computer

equipment to the data network access gateway. Also, high speed digital trunks interconnecting remote switches and routers of a data network are often leased from telephone long-haul carriers.

Nonetheless, telephone and data network infrastructures are usually
5 deployed together with limited sharing of resources, especially with regards to the core components of the networks--the switches and routers that steer the payloads throughout the networks. The cost of this redundancy coupled with advances in data network technologies has led, where possible, to integrated telephony traffic comprising voice, facsimile, and modem information over a
10 unified data network. As such, a data network dial-up access gateway should now be able to accept, service, and deliver any type of data over its access telephone links on a random, dynamic basis using a minimum set of hardware on a single platform.

In prior art multiservice processing applications involving the
15 processing of multiple data types, a matrix of digital signal processors (DSP) are typically required to perform digital signal processing (DSP) operations on a number of channels of data. For modem and facsimile traffic, the DSPs are mostly used to modulate and demodulate the traffic to and from the dial-up telephone access links. For a voice call over the same link, the same DSP can
20 instead be used to compress and decompress the voice traffic towards and from the core of the data network and to suppress undesirable echoes which usually arise at various points in the network.

Within the access gateway equipment, a host bus and host processor typically communicate payload data between the DSP processors of the array
25 and the data network side of the DSP array. A number of serial links typically communicate payload data between the DSP processors of the array and the telephone dial-up side of the array. The payload data can be one of several

types, and several processing routines, or algorithms, are typically required of each processor on a dynamic basis, where particular processing algorithms correspond to particular data types. Therefore, different algorithms which are resident as different software images are required to be loaded and run on

5 each of the DSP processors of the processor matrix on a dynamic basis. The problems inherent in dynamically loading and running different software images depends upon the prior art implementation selected on a platform. In spite of the presence of a corresponding separate external memory with each DSP of the processor array, it is prohibitive for this multiply instantiated

10 memory to be large enough to hold all the possible software modules necessary to process all data types.

→ One prior art DSP platform holds resident in a large memory of a host processor all algorithms necessary to process all data types serviced by the access gateway. Data received into this DSP platform is provided to a DSP

15 processor of the array. The DSP processor identifies the received data type and makes a request over to a host processor for the necessary processing software routine. The host processor responds by downloading the requested software routine to the DSP processor over the host bus.

The problem with this prior art implementation is that the host

20 processor and its associated bus are already heavily tasked in moving payload data into and out of the DSP processor array. Peak saturating bandwidths occur when many software download requests are responded to simultaneously. A typical example shows that bus bandwidth saturation is reached when the host processor services requests from five or more DSP

25 processors simultaneously. Therefore, the presence of 24 or 30 DSP processors in a multiservice access gateway servicing a T1 or E1 data line, respectively, would saturate the host bus and host processor. The occurrence of peak

saturating bandwidths results in an overall reduction in service reliability by the access gateway. Therefore, this prior art system is not upwardly scalable to include many DSP processors in the DSP processor array without a significant decrease in service reliability. Furthermore, an adverse impact on the
5 perceived quality of the delivered voice traffic would result.

Another prior art DSP platform uses larger shared memory systems among the members of small DSP processor groupings. Data received into this DSP platform is provided to a DSP processor of the array. The DSP processor identifies the received data type and accesses the shared memory
10 system of its grouping for the necessary processing software routine.

The problem with this prior art implementation is that a large memory system would have to be instantiated several times on the hardware platform because the number of processors sharing a software library would typically be required to be kept small for performance reasons. The performance of the
15 individual array processors is not reliable because the individual processing members of the array may periodically have to cease processing when accesses to the shared memory system are delayed because the memory system is responding to requests from other processors. Therefore, this implementation is inefficient with regard to cost, space, and performance.

20 An alternate prior art DSP platform holds resident in the private memory of each DSP processor of the array all of the required processing algorithms. Data received into this DSP platform is provided to a DSP processor of the array. The DSP processor identifies the received data type and processes the data using a software routine stored in its associated
25 memory.

The problem with this prior art implementation is that it requires significant duplication of memory assets. This results in a higher cost and the consumption of a significant amount of space in the hardware platform.

81862.P082

SUMMARY AND OBJECTS OF THE INVENTION

It is therefore an object of the invention to make available on a dynamic basis a large library of different firmware processing algorithms to each DSP processor engine of a DSP processor array.

5 It is a further object of the invention to efficiently perform digital signal processing of multiple types of data, using multiple firmware algorithms resident on a host platform, in order to communicate data over an Internet Protocol (IP) network and a public switched telephone network (PSTN).

10 It is a further object of the invention to increase the digital signal processing density when processing multiple types of data, using multiple firmware algorithms resident on a host platform, in order to communicate data over an Internet Protocol (IP) network and a PSTN.

15 It is a further object of the invention to reduce the memory, and corresponding platform area and space, required for performing digital signal processing of multiple types of data, using multiple firmware algorithms resident on a host platform, in order to communicate data over an Internet Protocol (IP) network and a PSTN.

20 It is a further object of the invention to reduce the cost per channel of digital signal processing when communicating multiple data types over an Internet Protocol (IP) network and a public switched telephone network (PSTN).

25 These and other objects of the invention are provided by a continuous broadcast of a number of firmware algorithms from a master DSP engine resident in a host processor to a number of service DSP engines of a DSP array over a channelized serial bus. The firmware algorithms are resident in a memory controlled by the master DSP engine.

In one embodiment, each service DSP engine is coupled to receive PCM data from multiplexed lines of a public switched telephone network and packetized data from an Internet Protocol (IP) network. The data may include, but is not limited to, modem data, voice data, audio data, video data, and facsimile data. The data is provided to one of a number of service DSP engines of a DSP array. Upon receipt of the data, each service DSP engine determines a data type of the received data and determines a firmware algorithm required to process the data. The service DSP engine then determines an address of at least one channel of the channelized serial bus on which the required firmware algorithm is available and unmask a corresponding bit of an interrupt mask in the service DSP engine. In response to the receipt of an interrupt signal corresponding to the unmasked interrupt bit, the service DSP engine executes an interrupt service routine resulting in the receipt and storage of the corresponding firmware algorithm from the master DSP engine. The interrupt is asserted by the firmware broadcast DSP just before block boundaries of the played-out firmware and is used by the receiving DSPs to synchronize the code upload to the code ployout. The service DSP engine processes the received data using the received firmware algorithm. When the data received by the service DSP engine is PCM data received from the PSTN, the service DSP engine produces packetized data for communication over the IP network. When the data received by the service DSP engine is packetized data received from the IP network, the service DSP engine produces PCM data for communication over the PSTN.

Other objects, features, and advantages of the present invention will be apparent from the accompanying drawings and from the detailed description which follows below.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example and not limitation in the figures of the accompanying drawings, in which like references indicate similar elements, and in which:

5 **Figure 1** is a communications system comprising the multiservice processing system of one embodiment.

Figure 2 is a multiservice processing system of one embodiment.

Figure 3 is a DSP module of one embodiment.

10 **Figure 4** is a master DSP engine and associated memory of one embodiment.

Figure 5 is a hardware block diagram of a DSP module of one embodiment.

Figure 6 is a data flow diagram of a DSP module of one embodiment.

Figure 7 is the TDM interface timing for a first timeslot.

DETAILED DESCRIPTION

A multiservice processing system is provided wherein a continuous broadcast of a number of firmware algorithms from a master DSP engine to a number of service DSP engines using a channelized serial bus is provided.

- 5 The firmware algorithms are resident in a shared memory controlled by the master DSP engine. The multiservice processing array provided herein provides, on a dynamic and continuous basis, a large library of different firmware processing algorithms to each DSP processor engine of a DSP processor array. The limited private external memory of each service DSP
- 10 engine is dynamically overwritten with the relevant firmware module depending on the incident data call type. This multiservice processing array efficiently performs simultaneous digital signal processing of multiple types of data, using multiple firmware algorithms resident on a host platform, in support of packetized data communication over an Internet Protocol (IP)
- 15 network and PCM data communication over a public switched telephone network (PSTN), thereby reducing the time and memory required for performing the processing. Therefore, the continuous firmware delivery system allows service DSP engines to dynamically reload with the appropriate algorithm or firmware overlay or subsystem software to service a random
- 20 incident telephony call types. These call types may comprise video, modem, facsimile, and voice data encoded using various protocols.

- Figure 1** is a communications system 100 comprising the multiservice processing system 102 of one embodiment. The multiservice processing system 102 receives data from a PSTN 104 over a multiplexed line 106. The
- 25 data is received as pulse coded modulation (PCM) data streams, but the embodiment is not so limited. In one embodiment, the multiplexed line 106 comprises 24 multiplexed data lines, but the embodiment is not so limited.

The multiplexed line 106 may be a T1 data line, a T3 data line, or an E1 data line, but the embodiment is not so limited. The data types 108-116 received over the PSTN 104 comprise, but are not limited to, facsimile data 108, modem data 110, video data 112, audio data 114, and voice data 116, for example telephone data. The lines 128-136 over which the data 108-116, respectively, is provided may be multiplexed data lines, but the embodiment is not so limited. The multiservice processing system 102 processes the received PCM data and generates packets or cells comprising the processed data. The packets or cells are provided to an IP network 118 for transmission.

The multiservice processing system 102 also receives packetized data from the IP network 118. The data types received in packets or cells over the IP network 118 comprise, but are not limited to, facsimile data, modem data, video data, audio data, and voice data, for example telephone data. The multiservice processing system 100 unpacks the packetized data and generates PCM data streams comprising the data. The PCM data streams are provided to the PSTN 104 using the multiplexed line 106. The PSTN 104 distributes the PCM data streams to individual subscribers 108-116 of particular destinations using data lines 128-136. The data lines 128-136 may be multiplexed data lines, but the embodiment is not so limited.

Figure 2 is a multiservice processing system 200 of one embodiment. The multiservice processing system 200 comprises a DSP array or matrix comprising five DSP modules (DSPMs) DSPM 1-5, but the embodiment is not so limited. **Figure 3** is a DSP module DSPM 1 of one embodiment. Each DSP module comprises six service DSP engines 212-217 divided into two groups wherein a first group comprises service DSP engines 212-214 and a second group comprises service DSP engines 215-217.

The multiservice processing system 200 supports the flow of data from the PSTN to the IP network and the flow of data from the IP network to the PSTN using PCM stream lines 230 and 232 and a host bus 220 coupled to a central processor unit (CPU) 204. To support data flow from the PSTN to the IP network, PCM data is received from the PSTN and provided to the service DSP engines using the PCM stream lines 230 and 232. A Time Division Multiplexed (TDM) interface (not shown) receives data from the PSTN over a multiplexed line, but the embodiment is not so limited. The PSTN data is processed and packetized by the service DSP engines 212-217, and the packetized data is provided to the IP network by the CPU 204 using the host bus 220 and a parallel memory mapped interface (not shown).

In supporting the flow of data from the IP network to the PSTN, packetized data is received from the IP network and provided to the service DSP engines 212-217 by the CPU 204 using the host bus 220 and a parallel memory mapped interface (not shown). The packetized data is unpacked and processed by the service DSP engines 212-217 to form PCM data streams. The PCM data is provided to the PSTN over the PCM stream lines 230 and 232.

In one embodiment, each DSP module comprises a programmable logic device (PLD) 240 that couples a TDM serial port of each service DSP engine 212-217 to a channelized serial bus 222. The TDM serial port of the master DSP engine 208, or jukebox DSP, is coupled to each of the service DSP engines 212-217 of the DSP array using the PLD 240 of each DSP module and the channelized serial bus 222. A firmware server interface provides a path from the master DSP engine acting as a firmware image broadcast server on the carrier to all service DSP engines on the multiple DSPMs. The interface couples the TDM serial port of the master DSP engine to the TDM serial port

of all service DSP engines. The host processor 204 may comprise the master DSP engine, but the embodiment is not so limited.

The channelized serial bus 222 supports unidirectional communications from the master DSP engine 208 to each of the service DSP engines 212-217. In one embodiment, the channelized serial bus 222 comprises five lines, four lines of which provide eight channels over which firmware algorithms are continuously broadcasted to the service DSP engines 212-217, but the embodiment is not so limited. The fifth line of the channelized serial bus 222 comprises an interrupt line that is coupled between the master DSP engine 208 and an interrupt pin of each service DSP engine 212-217. This interrupt line communicates a serial interrupt signal that signals the service DSP engines 212-217 that a particular block of memory, or portion of a firmware algorithm, is about to be played out. As a different interrupt signal pulse and identifier precedes each firmware algorithm block, the service DSP engines 212-217 know when the particular firmware algorithms are going to be provided, and therefore, when to upload code from the channelized serial bus 222.

The firmware algorithms allow the service DSP engines 212-217 to act as the gateway between the PSTN and IP network domains. This gateway function, however, goes beyond the transparent transfer of data in that the various types of PSTN payloads are typically processed to achieve greater quality or save bandwidth on the packet domain. Echo cancellation, parametric and non-parametric voice coding, suppression of packet bandwidth utilization during voice silence, facsimile relay, and modem relay, are a few examples of the processing provided by the firmware algorithms, but the embodiment is not so limited.

Figure 4 is a master DSP engine 208 and associated memory 210 of one embodiment. The master DSP engine 208 may reside in the host CPU, but the embodiment is not so limited. The master DSP engine is coupled to an external memory. The external memory comprises a number of firmware algorithms that are used to process the data received by the multiservice processing system. The master DSP engine, as discussed herein, provides continuous and repetitive broadcasts of these firmware algorithms over the channelized serial bus.

The master DSP engine 208 provides five output signals 402-410 comprising a clock signal 402, a frame signal 404, an address signal 406, a data signal 408, and an interrupt signal 410, but the embodiment is not so limited. The frame signal identifies the boundary of each frame of eight 16-bit words. The clock signal is the synchronous reference to which the data, frame, and address signals are latched in and out of the DSP engines. The data signal carries the actual binary levels of the payload stream, the algorithm program code.

The address signal is an 8-bit serial encoded address word that is used to identify the channel number. In one embodiment, the position of a "1" bit in the binary address will be synchronized with the channel number. Thus, during the first timeslot the binary address will be "00000001"; during the second timeslot the binary address will be "00000010"; during the third timeslot the binary address will be "00000100"; during the fourth timeslot the binary address will be "00001000"; during the fifth timeslot the binary address will be "00010000"; during the sixth timeslot the binary address will be "00100000"; during the seventh timeslot the binary address will be "01000000"; and, during the eighth timeslot the binary address will be "10000000". Using this technique the service DSP engines use the address signal as the channel

identifier and are capable of configuring their TDM ports to perform receive fetches on one or more specified channels.

Figure 5 is a hardware block diagram of a DSP module 500 of one embodiment. This DSPM 500 comprises six service DSP engines 212-217, each consisting of a Texas Instruments TMS320LC542-50 DSP and two banks of 64Kx16 static random access memory (SRAM) having 128K words total, but the embodiment is not so limited. In one embodiment, a 50 million instruction per second (mips) DSP engine will process one or two data channels comprising voice and facsimile and modem channels, but the embodiment is not so limited. As discussed herein, communication to the host processor is provided through a host bus interface that links the host ports of the service DSP engines 212-217 to the host bus with the assistance of a PLD 240. Control messaging and payload packets are moved through this interface, but the embodiment is not so limited.

In one embodiment, the host bus interface is a 16-bit parallel asynchronous interface used to access the host ports of the six service DSP engines and the control registers in the PLD. The DSPM is addressed as a data byte wide device via a dedicated select line and six address lines. Furthermore, the DSPM electrical interface provides for 16 bits of data connection and 19 bits of address line connectors. The DSPM electrical interface also provides two select lines. One select line provides the actual select signal while the second select line allows stacking of DSPMs. The second select line represents a path to the select line of the stacked DSPM.

The DSPM interface is designed to connect up to four 2MBps of bidirectional TDM streams using a TDM interface. The TDM interface connects to the buffered serial ports of the service DSP engines. The DSPM of one embodiment ports a multiframe synchronization signal for each of these

four streams; the DSPM terminates two of these streams and is wired to support two multiframe synchronization signals.

An interrupt status register allows the host CPU to view the sources of interrupts coming from the DSPM. Interrupting sources may be polled through this register even if masked through an interrupt mask register. A logic "1" indicates that the source, the host interrupt line of the particular service DSP engine, is interrupting the host CPU. The interrupt mask register allows the host to mask the sources of interrupt coming from the DSPM. A logic "0" written to the corresponding bit will mask the given service DSP engine's host interrupt line. An unmasked interrupting source will result in the DSPM interrupt line going to the host CPU being asserted low until masked or cleared at the source, the particular service DSP engine.

In receiving PSTN data for communication over an IP network, data is received by the multiservice processing system from the PSTN comprising data sampled at 8,000 samples per second where each sample comprises 8 bits. This PSTN data is selectively provided to the service DSP engines of the DSP array over the PCM stream lines, as discussed herein. Upon receipt of the data, each service DSP engine determines the data type of the received data. Furthermore, each service DSP engine determines a firmware algorithm required to process the received data. The service DSP engine then selectively monitors the channelized serial bus for at least one firmware algorithm, wherein the firmware algorithm is used to process the received data. When the firmware algorithm monitored for is detected on the channelized serial bus, the service DSP engine receives the firmware algorithm.

The selective monitoring performed by the service DSP engine comprises the service DSP engine determining an address of at least one channel of the channelized serial bus on which the required firmware

algorithm is available. In one embodiment, the serial bus comprises eight channels, but the embodiment is not so limited. Upon determination of the channel address, the service DSP engine unmask a bit of an interrupt mask in the service DSP engine, where the unmasked bit corresponds to the address
5 of at least one channel of the serial bus on which the firmware algorithm is transmitted.

In response to the receipt of an interrupt signal corresponding to the unmasked interrupt bit, the service DSP engine executes an interrupt service routine. The interrupt service routine causes the corresponding firmware
10 algorithm to be received by the service DSP engine. Furthermore, the interrupt service routine causes the received firmware algorithm to be stored in the SRAM of the corresponding service DSP engine. The service DSP engine uses the stored firmware algorithm to process the PSTN data received
15 by the service DSP engine thereby packetizing the data received from the PSTN. The service DSP engine loads the packetized data into a buffer of the service DSP memory. The CPU monitors the status of the buffer via the host bus. When packetized data is detected in the buffer, the data is downloaded over the host bus to router circuitry. The router circuitry provides the packetized data to the Internet Protocol (IP) network.

20 In the case where a data stream changes from one data type to another data type mid-stream, for example, where a telephone call transmits voice data and facsimile data, the service DSP engine processing that call will recognize any changes in data type and, in response, selectively monitor for
25 and receive a firmware algorithm to use in processing the new data type. A call type can be signaled to the service DSP engines by the host CPU at call onset and it can change between start and finish such as, for example, a voice call changing to facsimile data. The service DSP engines are able to take

configuration instructions from the host CPU and are able to detect the call types as well. In the case of a voice call changing to facsimile, the DSP is able to detect the transition, load the appropriate firmware module, and start execution within a nominal time.

5 In processing data received from an IP network for communication over a PSTN, packetized data is received from the IP network and selectively provided to the service DSP engines of the DSP array over the host bus using the CPU, as discussed herein. Upon receipt of the data, each service DSP extracts the data from the packet or cell used for transmission over the IP
10 network. The service DSP then determines the data type of the received data. Furthermore, each service DSP engine determines a firmware algorithm required to process the received data. The service DSP engine then selectively monitors the channelized serial bus for at least one firmware algorithm, wherein the firmware algorithm is used to process the received data. When
15 the firmware algorithm monitored for is detected on the channelized serial bus, the service DSP engine receives the firmware algorithm.

 The selective monitoring performed by the service DSP engine comprises the service DSP engine determining an address of at least one channel of the channelized serial bus on which the required firmware
20 algorithm is available. In one embodiment, the serial bus comprises eight channels, but the embodiment is not so limited. Upon determination of the channel address, the service DSP engine unmask a bit of an interrupt mask in the service DSP engine, where the unmasked bit corresponds to the address of at least one channel of the serial bus on which the firmware algorithm is
25 transmitted.

 In response to the receipt of an interrupt signal corresponding to the unmasked interrupt bit, the service DSP engine executes an interrupt service

routine. The interrupt service routine causes the corresponding firmware algorithm to be received by the service DSP engine. Furthermore, the interrupt service routine causes the received firmware algorithm to be stored in the SRAM of the corresponding service DSP engine. The service DSP engine uses the stored firmware algorithm to process the data received by the service DSP engine thereby generating PCM streams for communication to subscribers over the PSTN. The service DSP engine communicates the PCM streams over the PCM lines of the multiservice processing system to the multiplexed lines of the PSTN.

Figure 6 is a data flow diagram of a DSP module of one embodiment. The service DSP engines 212-217 of the DSP module are divided into two groups 610 and 620 with group 610 comprising service DSP engines 212-214 and group 620 comprising service DSP engines 215-217. This data flow scheme supports the flow of data from the PSTN to the IP network and the flow of data from the IP network to the PSTN. To support data flow from the PSTN to the IP network, PCM data is received from the PSTN and provided to the service DSP engines 212-217 using the PCM stream lines 230 and 232. The PSTN data is processed and packetized by the service DSP engines 212-217, and the packetized data is provided to the IP network over the host bus 220 using a parallel memory mapped interface. To support data flow from the IP network to the PSTN, packetized data is received from the IP network and provided to the service DSP engines 212-217 using the host bus 220 and the parallel memory mapped interface. The packetized data is unpacked and processed by the service DSP engines 212-217 to produce PCM streams. The PCM streams comprising the PCM data are provided to the PSTN over the PCM stream lines 230 and 232.

As discussed herein, the data flow scheme supporting data flow between the PSTN and the DSP module involves PCM formatted data flowing between the PSTN domain and the service DSP engine domain in PCM streams. The PCM streams are interfaced to the service DSP engines 212-217 through a buffered serial port. The PCM stream 230 carries PCM data to and from service DSP engines 212-214 for processing, while the PCM stream 232 carries PCM data to and from service DSP engines 215-217 for processing. Each of the PCM streams 230 and 232 comprise 32 channels per frame provided over a set of bidirectional communication lines; **Figure 7** is the TDM interface timing for the first timeslot of the 32 timeslots. The ingress lines of PCM stream 230 and 232 provide PSTN data to the service DSP engines 212-214 and 215-217, respectively, for processing in order to packetize the PSTN data for communication over an IP network. The egress lines of PCM stream 230 and 232 provide PCM data to the PSTN from service DSP engines 212-214 and 215-217, respectively, for communication to PSTN subscribers. The egress lines of PCM stream 230 and 232 provide a private frame pulse for each service DSP engine, but the embodiment is not so limited.

As further discussed herein, the data flow scheme supporting data flow between the IP network and the DSP module involves packetized data flowing between the IP network domain and the service DSP engine domain in packets or cells. The host bus, in supporting two-way communications, carries packetized data to and from service DSP engines 212-217 for processing. Consequently, the host bus provides packetized data to the service DSP engines 212-217 from the IP network for processing in order to generate PCM data for communication over the PSTN. The host bus also provides PSTN

data that has been packetized from the service DSP engines 212-217 to the IP network for communication.

As previously discussed for one embodiment, the continuous serial firmware broadcast system is implemented by designing onto the carrier card an extra DSP comprising a private memory large enough to store all the firmware modules required to process all of the data types processed by the multiservice processing system. This extra DSP, the jukebox DSP or master DSP engine, broadcasts firmware blocks continuously and repetitively out of a TDM serial port over the channelized serial bus to the TDM serial ports of all service DSP engines of the DSP array. Therefore, all DSP engines of the multiservice processing system are bussed together using the serial bus. As only the master DSP engine transmits on this bus, the service DSP engines receive communications from the master DSP engine, but the embodiment is not so limited. The service DSP engines, therefore, selectively receive particular firmware modules from the repetitive broadcast stream that are needed to process received data and ignore the broadcast of all other firmware modules. It is noted that the TDM serial ports of the DSP engines may be different from the buffered serial port through which PCM streams are interfaced to the service DSP engines, but the embodiment is not so limited.

Furthermore, it is noted that it is possible to use the TDM serial port inter-DSP communication facility among the six service DSP engines on any DSPM in a conventional way. This is done by isolating, using CPU control, the TDM port signal bus of the DSPM from its external source at the PLD; a bit in a Miscellaneous Control Register of the DSPM PLD controls this feature. When isolated from the master DSP engine signals, all service DSP engines within that particular DSPM can transmit on the inter-DSP communication link to other service DSP engines within that DSPM.

The firmware algorithms are sectioned and delivered to the service...

DSP engines using small fixed size blocks so that a service DSP engine in need of a module will not have to wait for the playout of that particular algorithm to begin again after just missing the beginning of the playout. As such, the beginning words of each block comprise a code that identifies that particular block. A service DSP engine monitoring the broadcast for a particular algorithm evaluates these identifier words in order to decide if a given block contains a portion of all of the algorithm for which the service DSP engine is monitoring. This technique allows acceptance of an algorithm to begin at multiple points during any given module playout as opposed to beginning only at the beginning of a module. Thus, this technique reduces the worst-case delivery time by nearly fifty percent.

The beginning blocks are signified to the service DSP engines by a broadcasted interrupt signal from the master DSP engine. The interrupt signal notifies the service DSP engines that the time frame pulse will correspond to the beginning of the first word of a new block. As this signal is connected to an interrupt input pin on each service DSP engine, a service DSP engine can mask the corresponding interrupt to ignore the broadcast or it can unmask the corresponding interrupt at the time it begins monitoring for a particular algorithm.

An interrupt service routine associated with a particular interrupt reads the next word received into the TDM port, the first word of the block, and calls the appropriate software routine to load the remainder of the block, if appropriate. If the identified block is not the block for which the service DSP has been monitoring, the software routine will simply re-enable the interrupt so that the following block can also be identified. This process will

repeat until all blocks of a module have been loaded, at which time the interrupt is remasked.

In one embodiment, a service DSP engine in need of a firmware algorithm module can be signaled by the CPU, using the host port communication, on which channel to find the module. This technique may be used in the case of CPU-initiated firmware configurations. Furthermore, in the case of service DSP engine-detected configuration requirements, the service DSP engine can use hard-coded information comprising the channel number of a particular firmware module.

The various firmware algorithms have different delivery time requirements that can be accommodated by using different delivery bandwidth allocations on the channelized serial link. For example, the facsimile relay algorithm, with the most critical delivery time requirement, is allocated one or more channels while algorithms with less critical delivery time requirements can be grouped together on a single channel. This allocation is achieved by appropriate distribution of the eight TDM channels of the TDM port interface among the different algorithms.

The invention has been described in conjunction with the preferred embodiment. Although the present invention has been described with reference to specific exemplary embodiments, it will be evident that various modifications and changes may be made to these embodiments without departing from the broader spirit and scope of the invention as set forth in the claims. Accordingly, the specification and drawings are to be regarded in an illustrative rather than a restrictive sense.